1. Stomeo E., T. Kalganova and C. Lambert (2006) Chose the Right Mutation Rate for Better Evolve Combinational Logic Circuits. Int'l Journal of Computational Intelligence (IJCI), ISSN 1304-2386, Vol. 2, No. 4, pp.286-277

Chose the Right Mutation Rate for Better Evolve Combinational Logic Circuits

Emanuele Stomeo, Tatiana Kalganova, Cyrille Lambert

***Abstract*—**Evolvable hardware (EHW) is a developing field that applies evolutionary algorithm (EA) to automatically design circuits, antennas, robot controllers etc. A lot of research has been done in this area several different EAs have been introduced to tackle numerous problems, as scalability, evolvability etc. However every time a specific EA is chosen for solving a particular task, all its components, such as population size, initialization, selection mechanism, mutation rate, and genetic operators, should be selected in order to achieve the best results. In the last three decade the selection of the right parameters for the EA's components for solving different "test-problems" has been investigated. In this paper the behaviour of mutation rate for designing logic circuits, which has not been done before, has been deeply analyzed. The mutation rate for an EHW system modifies the number of inputs of each logic gates, the functionality (for example from AND to NOR) and the connectivity between logic gates. The behaviour of the mutation has been analyzed based on the number of generations, genotype redundancy and number of logic gates for the evolved circuits. The experimental results found provide the behaviour of the mutation rate during evolution for the design and optimization of simple logic circuits. The experimental results propose the best mutation rate to be used for designing combinational logic circuits. The research presented is particular important for those who would like to implement a dynamic mutation rate inside the evolutionary algorithm for evolving digital circuits. The researches on the mutation rate during the last 40 years are also summarized.

***Keywords*—** Design of logic circuit, evolutionary computation, evolvable hardware, mutation rate.

I. Introduction

EVOLUTIONARY design of circuits, which is a branch of evolvable hardware [1–3], refers to a technique introduced to automatically design circuit where the circuit configuration is carried out by evolutionary algorithm (EA). The basic schema of an evolvable hardware (EHW) system is given in Fig. 1. The evolutionary algorithm provides the circuit configurations to the reconfigurable hardware, which could be an FPGA, FPTA or other customized chips. The electronic chip configures itself with the circuit configuration received and sends the circuit's response back to the evolutionary algorithm. Based on the response received the EA modifies the chromosome and supplies a new circuit configuration to the

Manuscript received October 25, 2005. This work was supported in part by the EPSRC under grant number GR/S17178/.

E. Stomeo, C. Lambert and T. Kalganova are with Brunel University, West London. UB8 3PH, Uxbridge, Middlesex, UK. (Tel: 0044 01895 266777; e-mail: emanuele.stomeo@brunel.ac.uk).

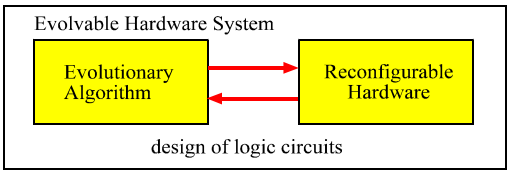


Fig. 1. Basic evolvable hardware system.

chip. EHW is a technique inspired by natural evolution [4]. These techniques began to be treated with increasing interest since the 60s when Holland introduced the concept of genetic algorithms (GA) [5], [6], which are the most general methods of solving search and optimization problems. A lot of research has been done in order to improve the classic GA for a given problem and many others evolutionary algorithms have been introduced as genetic programming (GP) [7], evolution strategy (ES) [8–11], evolutionary programming (EP) [12], [13], Cartesian Genetic Programming [14] etc. However, every time a specific evolutionary algorithm is chosen for solving a particular problem, all its parameters such as population size, type of initialization, selection mechanism, and genetic operators should be tuned in order to achieve the best results. This is because the efficiency of EA is highly dependent on all its parameters as already demonstrate by several researchers in [4], [15–19]. In order to find the best values for evolutionary algorithm's parameters several researchers have tuned them [18–20] in an attempt to find a general optimum for a set of test functions. However, the results obtained are different for different types of algorithms and problems as shown in Table 1. The design of circuits was not included in this set of test functions. Therefore the behavior of the mutation rate for designing combinational logic circuits has to be investigated. The mutation operation accomplishes simple operation which involves in flipping the value of some genes. The aim of this operation is to bring more change (diversity) into the population. By increasing the mutation rate, the genetic search will be transformed into a random search but it also helps to reintroduce lost genetic material [27]. In designing of combinational logic circuits change some genes inside the chromosome means to change the functionality of logic gates, for example from AND to XOR, and to change the connections between them.

The performance of the evolutionary algorithm (number of generations required to completely design the logic circuits) together with the quality (based on the value of the redundancy

TABLE 1. RESEARCH RESULTS ON MUTATION RATE

Author	Year	Approach	Proposed mutation rate	Problems
De Jong [19]	1975	GA (for online and offline performance)	0.001	General optimization problems (EHW not included)
Grefenstette [18]	1986	Meta GA	0.01	
Shaffer et al [21]	1989	GA (using online average performance)	0.005-0.01	Multimodal functions, FIR filter, 30 city travel sales person, graph partitioning
Mühlenbein et al. [22]	1992	Iterated Hillclimbing or (1+1,*m*,*hc*)-algorithm	$1/l$ l=chromosome length	Binary functions
Srinivas et al. [23]	1994	AGA	$0.5(f_{max}-f)/(f_{max}-f_{avg})$ where f_{max} is the maximum fitness value and f_{avg} is the average of the fitness	Several multimodal function including TSP, neural network weight optimization problems and generation of test vectors for VLSI circuits
Niwa et al [24]	1995	GA	$1/2n$ n=population size	Markov chain
Haupt [25]	2000	GA	0.05-0.2	Electromagnetic (array factors)
Nijssen [26]	2003	(1:λ) EA	$1/l$ l=bit-string length	Trap functions

and number of logic gates used during design and optimization of the logic circuits) of the obtained results has been studied for different values of mutation rate.

The experimental results achieved indicate that a fixed mutation rate should not be used for designing logic circuits. Furthermore the behavior of the mutation rate to be used during evolution, for those who want to use a dynamic mutation rate for design and optimization of logic circuits, has been extrapolated. In this paper we focus only on online average performance [21]. The (1+λ) evolution strategy already tested for its performance [35–36] has been chosen as evolutionary algorithm.

The paper is organized as follows: Section II gives a classification of the evolvable hardware systems and explains why an extrinsic evolvable hardware system (firstly introduced by H. de Garis [3]) has been chosen for the simulations. Section III describes an extrinsic evolvable hardware system, from the definition of the evolutionary algorithm to the description of the fitness functions implemented. Section IV gives the system set-up for the EA used. Section V presents the experimental results. Section VI provides a discussion of the results found. Last section gives conclusions and indicates possible areas for future investigation.

II. CLASSIFICATION IN EVOLVABLE HARDWARE

As proposed by Torresen [40], Andersen [41] and Gordon and Bentley [42], evolvable hardware can be classified in several classes, depending on: evolutionary algorithm, target technology, level of abstraction (Hirst [43]) and fitness evaluation. Based on that classification a simpler categorization is:

- Extrinsic environment [21], [28–32]
- Intrinsic environment [33], [34].
- Mixtrinsic environment [39].

Extrinsic EHW refers to a system whereby the evolutionary algorithm runs in software. Intrinsic EHW describes situations where the evolutionary algorithm is implemented in hardware and mixtrinsic evolvable hardware is a hybrid combination of intrinsic and extrinsic methods, usually the evolutionary algorithm with the fitness function evaluation is done in software and the evolved target implemented into hardware. The extrinsic evolvable hardware has been chosen as an object of investigation due to its ability to collect any relative information fairly easy.

III. EXTRINSIC EVOLVABLE HARDWARE

In this section the evolutionary algorithm used to evolve logic circuits, together with the fitness function and chromosome representations are presented. The approach in question was introduced in [28].

A. Evolutionary Algorithm

The evolutionary algorithm chosen for the evolution of combinational logic circuits is the (1+λ) evolution strategy already tested for its efficiency in [45]. In this approach λ represents the population size.

Each individual of the population represents a potential solution to a given task. The algorithm is very simple and easily implementable. In the first step all the chromosomes are randomly initialized. At the second step the fitness function of each individual is calculated (at each individual is assigned a fitness value according to how good it is), the fittest individual is selected and duplicated for the population to the next generations. The new population is brought up to date by mutating the best chromosome of the previous generation (see Fig. 2).

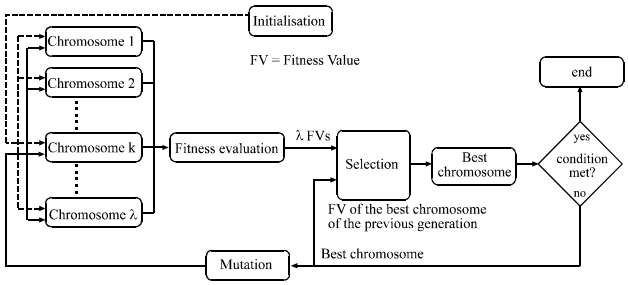


Fig. 2. Schematic of (1+λ) evolution strategy.

B. Chromosome Coding

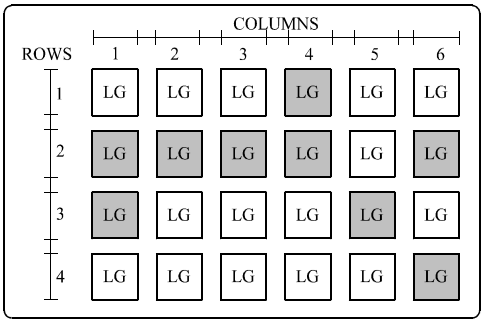
The chromosome is a string of parameters (known as *genes*) joined together. The string of genes represents a potential solution to a given problem. In evolutionary design of electronic circuits the chromosome contains all the needed information to describe the structure and the connectivity of the evolved combinational logic circuits. Since the final scope of this research is to design combinational logic circuits using Field Programmable Logic Array (FPGA), we have decided to represent the electronic circuits as a rectangular array of logic gates, see Fig. 3. The logic gates used for the simulations are: AND, OR, XOR, NOT with up to 4 inputs and MUX, where MUX is a multiplexer with 2 inputs and one control signal. The chromosome is divided in three components:

- Geometry which contains information about the *number of rows*, the *number of columns* of the rectangular array and the degree of internal connectivity, also referred to as *level-back parameter* [14]. The level-back parameter, or so called connectivity parameter, defines how many columns of cells to the left of the current column might have their outputs connected to the inputs of the current cell. The chromosome at this level is made of an array of 3 cells, the first contains the number of row, the second the number of columns of the rectangular array and the third contains the level back parameter.

- Functionality which describes the array of cells and determines the circuit's outputs. It is an array which identifies all the logic gates together with their functionality. The last cells of this array identify the logic gates from which the circuit's output are taken.

- Routing which represents the structures of each cell in the circuit and the connections between them. This is characterized by an array where the first cell identifies the logic gate, the second identifies it's the number of inputs and the other cells identify the connections with other logic gates.

The next session presents an example of chromosome encoding.



LG=Logic Gate randomly chosen from AND, OR, XOR, NOT and MUX.

LG LG=Logic Gate not used during current configuration.

LG LG=Logic Gate used during current configuration.

Fig. 3. Example of chromosome during evolution.

C. Example of Chromosome Coding

In this section an example of the chromosome encoding is presented. Supposing that for a particular experiment a circuit layout with 2 rows, 3 columns and the maximum internal connectivity (level back) is chosen. Therefore the chromosome that describes the geometry of the circuits is the following array: (2, 3, 3). The chromosome at functionality level is an array which identifies each logic gates together their functionality, based on the encoding table reported in Table 2, and the circuit's output. Therefore the chromosome at functionality level for the circuit reported in Fig. 4 is the following array: (**4**, 7, **5**, 8, **6**, 7, **7**, 9, **8**, 7, **9**, 8, *5*, *6*), see Fig. 5. The numbers in bold identify the logic gate, the number beside them identifies the functionality of that particular logic gate and the last number is italic identify form which logic gates the output of the circuits are taken. For example the first number "**4**" identifies the logic gate. The second number "7" identifies the functionality, AND gate in this case. The chromosome at routing level contains information regarding the structure of the logic gates and the connectivity between them. In relation of the circuit in Fig. 4, the chromosome at routing table is shown in Fig. 6. The first part of that array is (4, 2, 1, 2); the number "4" identifies the logic gate inside the circuit layout; "2" means that the logic gates identified by the number "4" has got two inputs which are taken from the output's gates "1" and "2".

TABLE 2. GENE FUNCTIONALITY

Gene functionality	Gate function
2	NOT
6	WIRE
7	AND
8	OR
9	XOR
12	Multiplexer

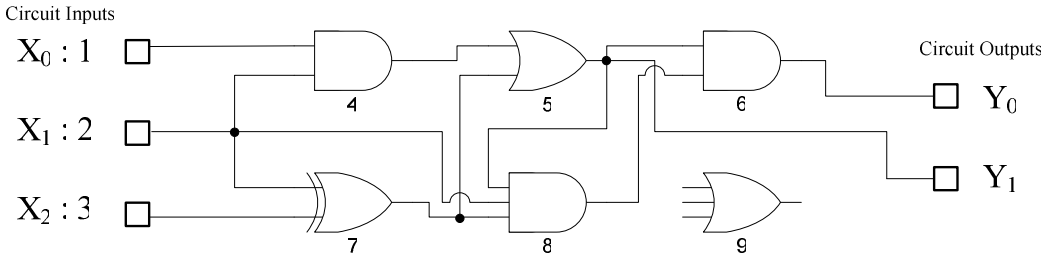


Fig. 4. Example of circuit

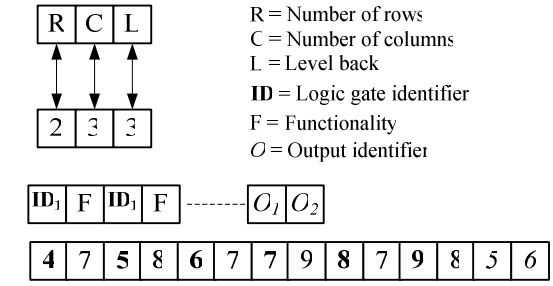


Fig. 5. Chromosome at geometry and functionality level.

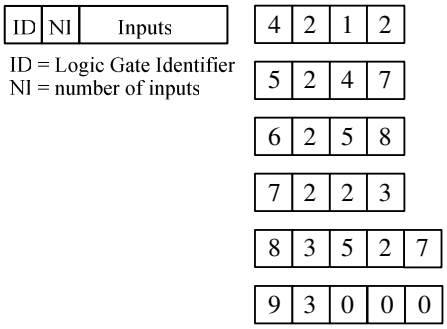


Fig. 6. Chromosome at routing level.

D. Redundancy

In this paper the redundancy of the combinational logic circuits is given by the Equation 1, where N_{LG} represents the total number of logic gates in our chromosome. N_{ALG} refers to the number of active logic gates, which are the logic gates that are currently used in a circuit configuration; therefore N_{ALG} is equal to or less than N_{LG}.

$$r = 1 - \frac{N_{ALG}}{N_{LG}} \tag{1}$$

Let us consider a numerical example, considering the circuit layout in Fig. 3. The rectangular array is made of 6 columns and 4 rows of logic gates. The highlighted logic gates are used for a particular configuration; therefore they are connected to each other in order to create the logic circuit. The logic gates not highlighted are not connected, thus they are redundant. In this example, N_{LG} (the total number of logic gates) is 6*4=24. The number of active logic gates (the logic gates that participate in creating the digital circuit) N_{ALG} is 9. The redundancy, calculated using equation (1) for the circuit's configuration of Fig. 3, is 0.625.

E. Fitness Function

In evolvable hardware the fitness function evaluates the evolved circuits in terms of their functionality. Given a particular chromosome the fitness function returns a value which is supposed to be proportional to the utility and ability of the individual which that chromosome represents [44]. In our experiment a multi-objectives fitness function has been considered. It has two main criteria: first *design* the fully functional circuit and second, once the circuit is fully functional evolved, *optimization* which leads to reduced numbers of logic gates used in the circuit configuration.

The fitness function f is calculated as:

$$f = \begin{cases} f_1 & f < 100 \quad \text{circuit design} \\ f_1 + f_2 & f \geq 100 \quad \text{circuit optimization} \end{cases} \tag{2}$$

where f_1 is a design criterion that defines the percentage of correct bits in the evolved circuit, f_2 is the optimization criterion for the optimization stage. The fitness function for the functionality of the evolved circuit f_1 is calculated as follows:

$$f_1 = 100 - \frac{100}{m \cdot p} \sum_{f_c=0}^{2^n-1} \sum_{i=0}^{m-1} \left| y_i - d_i \right| \tag{3}$$

where m and n are the number of outputs and the number of inputs of the given logic function, respectively; p is the number of input-output combinations; y_i is the i^{th} digit of the output combination produced by the evaluation of the circuit, d_i is the desired output for the fitness case f_c. $|y_i-d_i|$ is the absolute difference between the actual and the required outputs.

The fitness function for the optimization stage has been calculated below, where N_{LG} is the number of the total logic gates present in the chromosome, so N_{LG} is equal to the number of rows multiplied by the number of columns of the

chromosome. N_{PLG}^{max} is the number of primitive logic gates necessary for building the logic gate with the highest number of inputs present inside the chromosome. Fig. 7 shows how to decompose a logic gates with 4 inputs. Therefore if a logic gate has 4 inputs the number of primitive logic gates necessary to build it is 3. N_{row} and N_{col} are the number of rows and columns of the chromosome. $N_{PLG_{(i,j)}}$ is the number of primitive logic gates necessary to build the $(i, j)^{th}$ logic gates. $N_{PLG_{(i,j)}}$ is 0 if the $(i, j)^{th}$ logic gate is unconnected.

$$f_2 = N_{LG} \cdot N_{PLG}^{\max} - \sum_{i=1}^{N_{row}} \sum_{j=1}^{N_{col}} N_{PLG_{(i,j)}} \qquad (4)$$

Fig. 8 shows the behaviour of a fitness function during the evolution of a 2 bit multiplier. In that figure two different stages are noticeable. The first shows the design of the multiplier, with each generation the fitness function value increases until it reaches 100%. At this point the functionality of the circuit is completely evolved. During the first stage the fitness function is calculated based on equation 3. The second stage starts just after the circuit is evolved. This stage performs the optimization of evolved circuits by reducing with each generation the number of active logic gates. Furthermore during this stage the fitness function, calculated from equation 4, also increases its value because the circuit is better optimised.

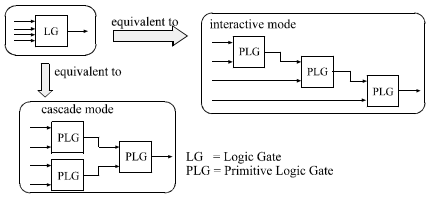


Fig. 7. Decomposition of a non primitive logic gate into primitive logic gates

IV. Setting Parameters

In this section the system set-up used to carry out all the experiments is described. Firstly the evolutionary algorithm's parameters used are given, and then the circuit layout and the logic of the evolved circuits are provided. Finally a description of the circuits evolved, together with one example, is given.

A. *Initial Parameters*

In Table 3 the evolutionary algorithm's parameters are given; where the number of generations refers to the number of cycles which each experiment has been evolved; population size refers to the number of different chromosomes; termination criteria is the maximum number of generations the evolutionary algorithm can perform before the process will be stopped; the mutation rate modifies the cell input, cell type (for example form AND to XOR) and circuit output.

For the given logic functions the results are considered only if the logic circuit has been successfully evolved 100 times out of 100 runs (i.e. success rate 100%).

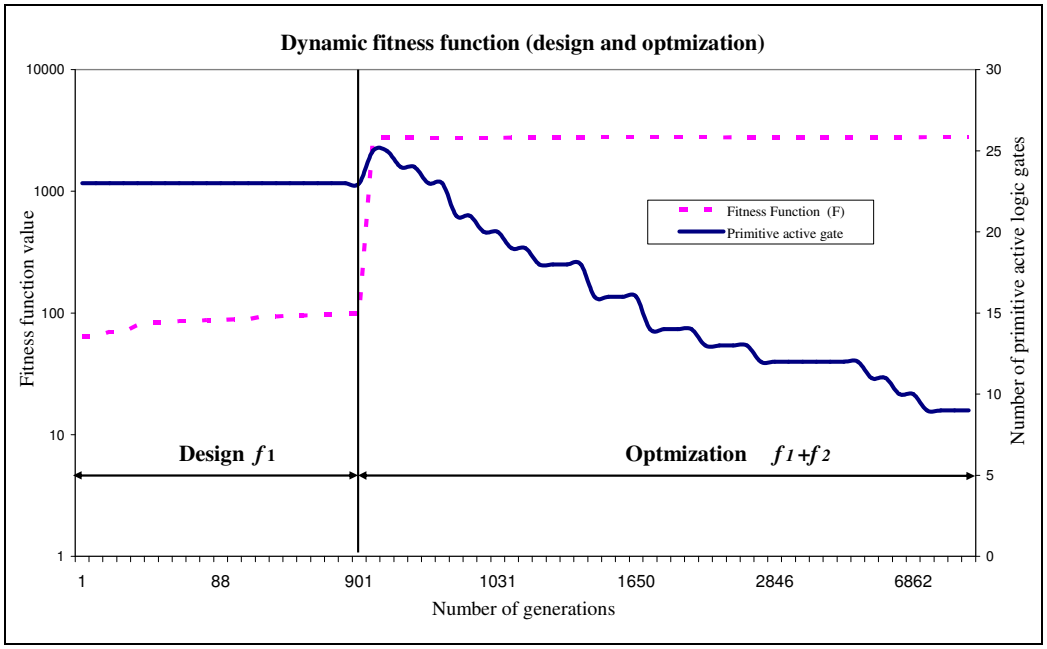


Fig. 8. This graphic shows the behaviour of the fitness function. Two different stages may be seen, design f1 and optimization ($f_1 + f_2$). It is also notable that the number of primitive active gates is reduced.

TABLE 3. INITIAL DATA FOR THE EXPERIMENTS CARRIED OUT USING (1+λ) EVOLUTION STRATEGY.

Number of generations	5000
Population size (λ)	5
Number of runs per each evolved circuit	100
Termination criteria for evolutionary process	5000 generations
Mutation rate	0.09 – 0.5
Elitism	yes

TABLE 4. INITIAL DATA: DIMENSION SIZE AND CONNECTIVITY OF THE CIRCUIT LAYOUT USED DURING SIMULATIONS

Circuit layout	Number of rows	10
	Number of columns	10
	Level back or connectivity parameter [14]	10

```
.i 3
.o 3
.p 8
000 010
001 011
010 101
011 101
100 011
101 110
110 101
111 011
.e
```

Fig. 9. Example of truth table (in Berkley format) used for the evolution of logic circuit with (1+λ) evolution strategy.

Each logic circuit has been evolved 100 times for each different mutation rate, starting from 0.09 to 0.5 with an increasing step of 0.01. In Table 4 the features of the circuit layout are given. Definitions of the number of rows, columns and level back have been provided in the previous section. The logic gates that participate in evolutionary processes are AND, OR, XOR, NOT, and multiplexer with 2 inputs and one control.

The connection between building blocks (combination of primitive logic gates) is in interactive and cascade mode. Each logic gate has up to 4 inputs. The structures of cascade and interactive building blocks are given in Fig. 7. The logic circuits evolved are randomly generated and fully defined by truth tables. The truth tables used to describe the logic circuits are compatible with the Berkeley format, see Fig. 9. where .i specifies the number of inputs, .o the number of outputs, .p the number of product or input-output combinations and .e the end of file.

V. EXPERIMENTAL RESULTS

In this section the results of the evolved logic circuits are presented. The intention of these experiments is to analyse the variation of mutation rate influences:

- the number of generations required to design the circuit
- the number of active logic gates obtained during design
- the number of logic gates after the optimization stage
- the redundancy of logic gates

In Fig. 10 and Fig. 11 the average of the number of generations of the evolved logic circuits with 2 and 3 inputs respectively is given. This average is calculated by taking into account the results out of 100 runs, which are all successfully evolved. To better clarify how the average has been calculated. Let us consider the circuit with 2 inputs and 3 outputs see Fig. 10. This circuit has been evolved 100 times with mutation rate equal to 0.009; 100 times with mutation rate 0.01 and so on until the mutation rate is equal to 0.5. So, this circuit has been evolved 1500 times. The average is calculated out of 100 runs per each value of mutation rate. Therefore each point on that graph represents the average out of 100 runs of the number of generations required to evolve the circuits.

From Fig. 10 and Fig. 11 it may be observed that, in terms of the number of generations for evolving small circuits, the best value of mutation rate is 0.1.

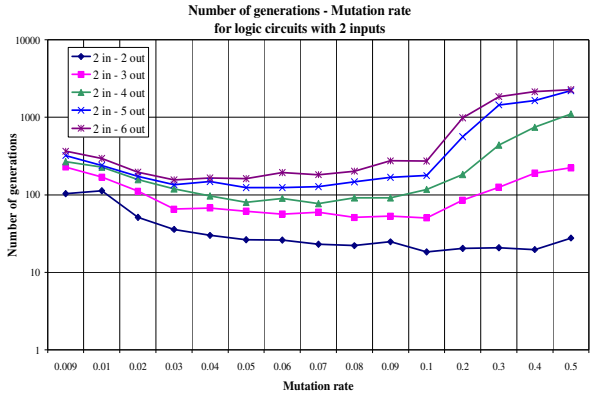


Fig. 10. Average out of 100 experiments of the number of generations required to completely evolve a logic circuit by changing the mutation rate. The circuits are with 2 inputs and varying numbers of outputs.

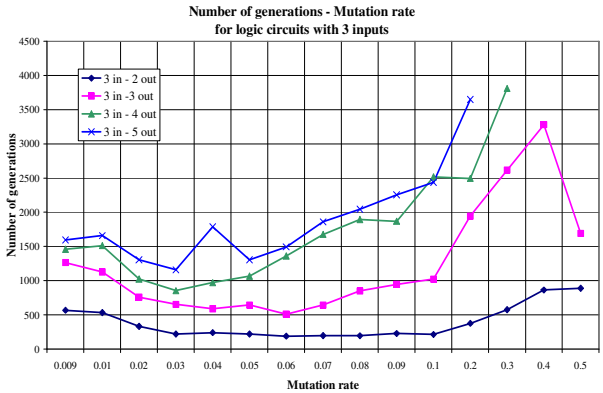


Fig. 11. Average out of 100 experiments of the number of generations required to completely evolve a logic circuit by changing the mutation rate for evolving logic circuits with 3 inputs. The solutions are given only for the circuits which are 100% evolved.

TABLE 5. SIMULATION RESULTS: REDUNDANCY OF EVOLVED RANDOMLY GENERATED LOGIC CIRCUITS. N.E. REFERS TO CIRCUITS WHICH ARE NOT EVOLVED WITH SUCCESS RATE EQUAL TO 100%. IN EACH CELL THE AVERAGE VALUES OF THE REDUNDANCY CALCULATED USING 100 EXPERIMENTS ALL COMPLETELY EVOLVED ARE REPORTED. THE BEST CIRCUIT'S CONFIGURATION FOR EACH LOGIC CIRCUIT IS HIGHLIGHTED.

Redundancy of evolved logic circuits																	
Logic circuits			**Mutation rate**														
in	**out**	**p**	**0.009**	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
2	**2**	**4**	0.816	0.830	0.809	0.824	0.824	0.829	0.811	0.826	0.824	0.829	0.829	0.819	0.851	0.840	0.842
2	**3**	**8**	0.762	0.759	0.750	0.771	0.752	0.758	0.769	0.777	0.776	0.769	0.772	0.801	0.812	0.816	0.817
2	**4**	**16**	0.711	0.729	0.717	0.710	0.700	0.731	0.742	0.730	0.744	0.748	0.756	0.777	0.779	0.809	0.789
2	**5**	**32**	0.671	0.686	0.683	0.687	0.692	0.700	0.703	0.703	0.720	0.713	0.700	0.723	0.765	0.763	0.749
2	**6**	**64**	0.646	0.644	0.650	0.675	0.674	0.665	0.672	0.668	0.680	0.668	0.691	0.715	0.730	0.734	0.748
3	**2**	**4**	0.790	0.788	0.790	0.809	0.808	0.798	0.813	0.814	0.811	0.818	0.816	0.845	0.850	0.868	0.869
3	**3**	**8**	0.738	0.735	0.726	0.740	0.740	0.740	0.740	0.752	0.748	0.775	0.781	0.805	0.837	0.831	0.737
3	**4**	**16**	0.685	0.688	0.691	0.704	0.700	0.707	0.709	0.738	0.732	0.723	0.733	0.802	0.755	N.E.	N.E.
3	**5**	**32**	0.635	0.657	0.655	0.676	0.681	0.677	0.693	0.687	0.706	0.703	0.694	0.690	N.E.	N.E.	N.E.
3	**6**	**64**	0.638	0.623	0.654	0.652	0.650	0.665	0.674	0.703	0.686	0.715	0.711	0.773	N.E.	N.E.	N.E.

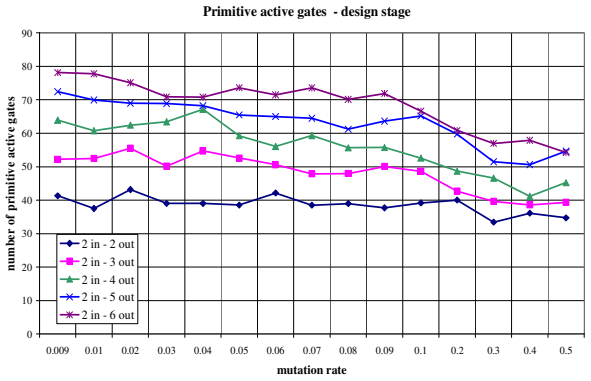


Fig. 12. Average out of 100 experiments of the primitive active gates required for the design stage, before the logic circuits are optimized. The solutions are given only for the circuits which are 100% evolved. The circuits are with 2 inputs and varying number of outputs.

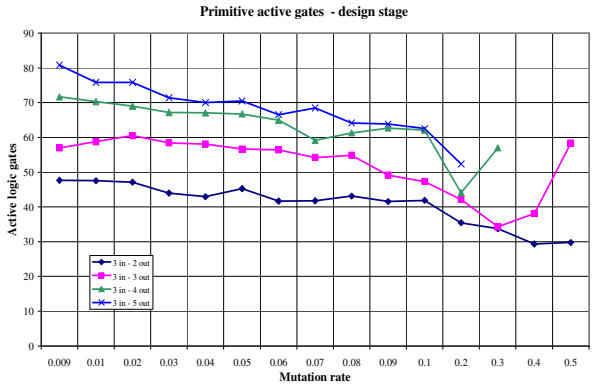


Fig. 13. Average out of 100 experiments of the primitive active gates required for the design stage, before the logic circuits are optimized. The solutions are given only for the circuits which are 100% evolved. The circuits are with 3 inputs and varying number of outputs

By increasing the complexity of the logic circuit based on the number of input-output combinations [38], the mutation rate which gives the best performance in terms of number of generations is decreased to 0.03. Therefore, the best mutation rate should be chosen according to the complexity of the task to be solved. The complexity of those circuits in evolvable hardware is mainly dependant on the number of inputs rather than outputs [38]. Moreover it may be noticed in Fig. 11, that if the mutation rate is very high (more than 0.3) the evolution of more complex tasks is not performed. This is because of the high randomness introduced in the chromosome. Therefore, the random processes become dominant under the evolutionary process if a mutation rate higher than 0.3 is used.

Table 5 illustrates the quality of the evolved logic circuits based on redundancy (before the optimization stage). The circuits with higher redundancy are those obtained with very high mutation rates. These results place us in contrast to the previously obtained results which advise the use of smaller values of mutation rate in order to find a solution with fewest numbers of generations. Therefore at this point the best trade-off between a small mutation rate for finding the final circuit's configuration with the fewest number of generations and a high mutation rate for finding the better-optimized circuits should be determined. However, before the final solution is outlined, the number of active logic gates required during design and optimization should be analyzed. Fig. 12 and Fig. 13 show the number of active logic gates by varying the mutation rate when the circuits are designed.

The experimental results demonstrate that the circuits with fewer logic gates are those created with high mutation rates, between 0.1 and 0.3. In Table 6 the number of active logic gates used after the optimization stage is presented. In that table the solutions with the smallest number of active logic gates are highlighted.

One may notice that the best solutions are achieved by decreasing the mutation rate as the complexity of the logic circuits increases.

TABLE 6. SIMULATION RESULTS OF THE MUTATION RATE FOR EVOLVING RANDOMLY GENERATED LOGIC. N.E. REFERS TO CIRCUITS WHICH ARE NOT EVOLVED WITH SUCCESS RATE EQUAL TO 100%. IN THIS TABLE THE BEST SOLUTIONS (THOSE WITH THE SMALLEST AMOUNT OF LOGIC GATES REQUIRED) ARE HIGHLIGHTED. IN EACH CELL THE AVERAGE VALUES OF THE REDUNDANCY CALCULATED USING 100 EXPERIMENTS IS REPORTED.

Average of number of active logic gates after optimization stage																
Logic circuits			**mutation rate**													
in	**out**	**p**	**0.009**	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
2	**2**	**4**	2.75	2.45	2.11	2.17	2.04	2.08	2.01	2.00	2.00	2.01	2.01	2.00	2.00	2.00	2.00
2	**3**	**8**	4.36	4.25	3.51	3.40	3.23	3.16	3.14	3.27	3.06	3.10	3.06	3.05	3.03	3.38	4.25
2	**4**	**16**	6.23	5.56	4.69	4.32	4.33	4.37	4.18	4.16	4.15	4.12	4.10	4.11	7.41	10.57	17.59
2	**5**	**32**	7.24	6.98	5.95	5.46	5.45	5.35	5.27	5.38	5.36	5.33	5.24	12.61	29.19	27.96	41.65
2	**6**	**64**	8.21	7.52	6.59	6.35	6.21	6.13	6.10	5.87	5.78	6.07	5.88	16.97	36.00	43.48	49.00
3	**2**	**4**	10.43	10.05	8.49	7.42	7.62	7.69	7.12	7.11	6.86	7.18	7.39	7.69	8.55	10.77	12.55
3	**3**	**8**	16.84	16.50	13.37	11.25	11.05	10.57	10.85	10.88	11.81	11.45	11.66	24.07	26.44	36.40	50.00
3	**4**	**16**	24.18	21.78	17.66	16.70	16.61	18.52	19.71	24.96	27.70	32.23	36.84	33.00	55.00	N.E.	N.E.
3	**5**	**32**	26.99	24.95	20.64	20.77	21.12	20.98	21.75	24.63	28.65	30.71	39.62	36.33	N.E.	N.E.	N.E.
3	**6**	**64**	26.82	29.54	22.11	22.13	26.15	29.09	27.75	33.75	36.86	36.61	37.38	47.33	N.E.	N.E.	N.E.

Based on those results one may conclude that the mutation rate for the better optimized circuit in terms of logic gates is inversely proportional to the complexity of the evolved circuit. Therefore, supposing that a very simple circuit should be solved, the best mutation rate to be chosen in order to have the best result in terms of number of logic gates should be between 0.3 and 0.5. If the circuits are more complex, that value (according with the experimental results shown in Table 6) should be reduced to between 0.02 and 0.04. By taking into account the results found in terms of number of generations, redundancy and number of active logic gates used for getting the optimized solutions, one may conclude that the mutation rate should be chosen according to the complexity of the task to be evolved and with the wishes of the user: less generations, fewer logic gates or good redundancy. Simpler tasks should be solved with a higher mutation rate. By increasing the complexity of the task the mutation rate should decreased to 0.02-0.04.

VI. DISCUSSION OF THE RESULTS

The behavior of the mutation rate observed is particularly important for those researchers wishing to implement a dynamic mutation rate inside the evolutionary algorithm for designing combinational circuits. It is also important for those who evolve large circuits [45] using decomposition strategies. For instance, supposing that a large circuit should be evolved; the mutation rate should be very low, as depicted above. When the stalling effect in the fitness function occurs (i.e. the evolutionary algorithm is not able to produce better results and the fitness values of the individuals of the population no longer increase) the system will be decomposed, usually using Shannon decomposition (see Fig. 14) [29] and the evolution of the circuit will continue with two or more simpler sub-circuits. At this stage the mutation rate should be changed in line with the complexity of those sub-circuits.

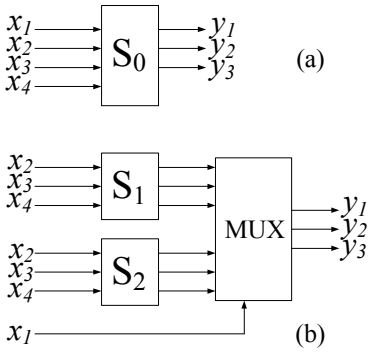


Fig. 14. Shannon decomposition.

VII. CONCLUSION

This paper describes how the mutation rate for an evolvable hardware system should be chosen in order to solve and better optimize the evolution of logic circuits. It should be noted that the mutation rate for EHW systems modifies the logic cell inputs, the cell functionality (for example from AND to NOR) and the system output. The experimental results found prove that the mutation rate should be inversely proportional to the complexity of logic circuits: more complex circuits require a smaller mutation rate. Therefore, these results are especially important for all researchers who are using decomposition strategies for evolving logic circuits, because they may now implement a dynamic mutation rate which changes in real-time, based on the complexity of the decomposed task. Further work will be focused on exploring the evolution of bigger logic circuits together with the use of different population sizes. This will be done in order to identify the best set up for all the parameters in the evolutionary algorithms for designing logic circuits.

ACKNOWLEDGMENT

First author thanks Dr Hemantha Kodikara-Arachchi for his valuable suggestions.

REFERENCES

[1] X. Yao, T. Higuchi; "Promises and challenges of evolvable hardware" *IEEE Trans. Systems, Man and Cybernetics, Part C*, volume 29, pp. 87 - 97, February 1999.

[2] H. de Garis. "Evolvable Hardware: Principles and Practice". *Communications of the Association for Computer Machinery (CACM Journal)*. August 1997.

[3] H. de Garis. "An Artificial Brain: ATR's CAM-Brain Project Aims to Build/Evolve an Artificial Brain with a Million Neural Net Modules Inside a Trillion Cell Cellular Automata Machine", *New Generation Computing J.*, 12, no. 2, pp. 215 – 221. 1994.

[4] D. E. Goldberg. Genetic algorithm in search, optimization and machine learning. Addison-Wesley Publishing Company, Incorporated, Reading, Massachusetts, 1989.

[5] J. Holland. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[6] M. D. Vose. "The Simple Genetic Algorithm". MA: MIT Press 1999.

[7] J. R Koza. Genetic Programming: On the Programming of Computers by Means of Natural selection. ISBN 0-262-11170-5. MIT Press, 1992.

[8] I. Rechenberg, "Evolution Strategy", in J. Zurada, R. Marks II, and C. Robinson (Eds.), *Computational Intelligence: Imitating Life*, 1994, pp. 147-159.

[9] H. G. Beyer and H. P. Schwefel, "Evolution strategies: A comprehensive introduction," *Natural Computing: an international journal*. Volume 1, Issue. 1, pp. 3–52, 2002.

[10] T. Bäck, Evolutionary Algorithms in Theory and Practice. New York: Oxford Univ. Press, 1996.

[11] H.-P. Schwefel, Numerical Optimization of Computer Models. New York: Wiley, 1981.

[12] L. J. Fogel, A. J. Owens, M. J. Walsh. Artificial Intelligence through Simulated Evolution. New York: Wiley, 1966.

[13] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Germany: Springer-Verlag, 1994.

[14] J. F. Miller and P. Thomson. "Cartesian genetic programming". In Riccardo Poli, Wolfgan Banzhaf, William B. Langdon, Julian F. Miller, Peter Nordin and Terence C. Forgaty, editors. *Genetic Programming, Proceedings of EuroGP 2000*. Vol. 1802 of LNCS, pages 121-132, Edinburg, 16 April 2000. Springer-Verlag.

[15] Myung-Sook Ko, Tae-Won Kang and Chong-Su Hwang. "Function optimisation using an adaptive crossover operator based on locality". *Eng. Applic. Artif. Intell.* Vol. 10, No 6 pp. 519-524, 1997.

[16] K. Y. Chan, M. E. Aydin, T. C. Fogarty; "Parameterisation of mutation in evolutionary algorithms using the estimated main effect of genes" *Congress on Evolutionary Computatio. CEC2004*. ,Volume: 2 , 19-23 June 2004 Pages:1972 – 1979.

[17] K. Y. Chan, M. E. Aydin, T. C. Fogarty; "An epistasis measure based on the analysis of variance for the real-coded representation in genetic algorithms" *Congress on Evolutionary Computation. CEC '03*. Vol.: 1 , 8-12 Dec. 2003 Pages:297 – 304.

[18] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Systems, Man, Cybernetics*. Vol. 16, no. 1, pp. 122–128, 1986.

[19] K. De Jong, "The analysis of the behavior of a class of genetic adaptive systems." Ph.D. dissertation, Dept. Computer Science, University of Michigan, Ann Arbor, 1975.

[20] A. E. Eiben, R. Hinterding, Z. Michalewicz; "Parameter control in evolutionary algorithms" *IEEE Transactions on Evolutionary Computation*, Volume: 3, Issue: 2, July 1999 Pages:124 – 141.

[21] J. D. Schaffer, R. Caruana, L. Eshelman and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization." *Proceedings of the Third International Conference on Genetic Algorithms*, ed. J. D. Schaffer, Los Altos, CA: Morgan Kaufmann, June 4-7, 1989, pp. 51-60.

[22] H. Mühlenbein. "How genetic algorithms really work: I.Mutation and Hillclimbing," in *Parallel Problem Solving from Nature- PPSN II*, R. Männer and B. Manderick, Eds., Amsterdam, The Netherlands, 1992, pp. 15-25.

[23] M. Srinivas, L. M. Patnaik. "Adaptive probabilities of crossover and mutation in genetic algorithms". *IEEE Transactions on Systems, Man and Cybernetics*, Volume 24, Issue 4, April 1994 Page(s):656 - 667

[24] T. Niwa, M. Tanaka. "On the mean convergence time for simple genetic algorithms". *IEEE International Conference on Evolutionary Computation*. Volume 1, 29 Nov.-1 Dec. 1995 Page(s):373.

[25] R. L. Haupt. "Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors". *IEEE International Symposium Antennas and Propagation Society*. Volume: 2, 16-21 July 2000. Pages:1034 – 1037

[26] S. Nijssen, T. Back; "An analysis of the behaviour of simplified evolutionary algorithms on trap functions". *IEEE Transactions on Evolutionary Computation.* Volume: 7, Issue: 1, Feb. 2003. Pages:11 – 22.

[27] M. Srinivas, L. M. Patnaik; "Genetic algorithms: a survey". *IEEE JNL Computer*, Volume: 27, Issue: 6, June 1994. Pages:17 - 26

[28] T. Kalganova, J. Miller, "Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness". *Proc. of the First NASA/DoD Workshop on Evolvable Hardware. IEEE Computer Society*, Pages 54–63. July 1999

[29] T. Kalganova; "Bidirectional incremental evolution in extrinsic evolvable hardware". *Proc. of the Second NASA/DoD Workshop on Evolvable Hardware. IEEE Computer Society*, 13-15 July 2000. Pages: 65 – 74.

[30] E. H. Luna, C.A. Coello Coello, A.H. Aguirre. "On the use of a population-based particle swarm optimizer to design combinational logic circuits". *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, 24-26 June 2004. Pages:183 – 190.

[31] S. Balkir, G. Diindar, G. Alpaydin,; "Evolution based synthesis of analog integrated circuits and systems" *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, 24-26 June 2004 Pages:26 – 29.

[32] M. Oltean, C. Grosan; "Evolving digital circuits using multi expression programming" *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, 24-26 June 2004 Pages:87 – 94.

[33] Yang Zhang, S.L. Smith, A.M. Tyrrell. "Digital circuit design using intrinsic evolvable hardware" *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, 24-26 June 2004 Pages:55 – 62

[34] J.C. Gallagher, S. Vigraham, G. Kramer; "A family of compact genetic algorithms for intrinsic evolvable hardware". *IEEE Transactions on Evolutionary Computation*, Volume: 8 , Issue: 2 , April 2004 Pages:111 – 126.

[35] J. Miller. "An empirical study of the efficiency of learning Boolean functions using a Cartesian genetic programming approach" *In Proc. of the Genetic and Evolutionary Computation Conference.* Volume 1, pp. 1135–1142, Orlando, USA, July 1999.

[36] T. Bäck, F. Hoffmeister, H. P. Schwefel. "A survey of evolutionary strategies". In R. Belew and L. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 2–9, San Francisco, CA, 1991. Morgan Kaufmann.

[37] H. P. Schwefel. Numerical Optimization of Computer Models. John Wiley & Sons, Chichester, UK, 1981.

[38] E. Stomeo and T. Kalganova. "Improving EHW performance introducing a new decomposition strategy." 2004 *IEEE Conference on Cybernetics and Intelligent Systems*. Singapore 1-3 December 2004, pp. 439-444.

[39] A. Stoica, R. Zebulum, D. Keymeulen. "Mixtrinsic Evolution". *In Fogarty, T., Miller, J., Thompson, A., Thompson, P. (Eds.), Proceedings of the Third International Conference on Evolvable systems: From Biology to Hardware (ICES2000)*, April 17-19, 2000, Edinburgh, UK. New York, USA, Springer Verlag, 208-217.

[40] J. Torresen. "Possibilities and Limitations of Applying Evolvable Hardware to Real-World Applications". *Proc. of the 10th International Conference on Field Programmable Logic and Applications*, Villach, Austria, pp. 230-239. 2000.

[41] P. Andersen P. "Evolvable Hardware: Artificial Evolution of Hardware Circuits in Simulation and Reality", M.Sc. Thesis, University of Aarhus, Denmark. 1998.

[42] Timothy G. W. Gordon and Peter J. Bentley. “On Evolvable Hardware”. *In Ovaska, S. and Sztandera, L. Soft Computing in Industrial Electronics*. Physica-Verlag, Heidelberg, Germany, pp. 279-323. 2002.
[43] A. J. Hirst. “Notes on the Evolution of Adaptive Hardware”, *Proc. of Adaptive Computing in Engineering Design and Control*, Plymouth, U.K., pp. 212-219. 1996.
[44] D. Beasley, D. R. Bull, R. R. Martin. “An Overview of Genetic Algorithms: Part 1, Fundamentals”. University Computing, 1993, 15(2) 58-69; ©Inter-University Committee on Computing.
[45] E. Stomeo et al. “On Evolution of Relatively Large Combinational Logic Circuits”. The 2005 NASA/DoD Conference on Evolvable Hardware. June 29 - July 1, 2005, Washington DC, USA. IEEE Computer Society. Pages 59-66.

Emanuele Stomeo received a Laurea degree in electronic engineering from Politecnico di Torino, Turin, Italy in 2003. He is currently working towards a PhD in computer science and engineering at Brunel University, West London, UK. From 2000 to 2003 he studied at RWTH Aachen University, Germany where he pursued specializations in image processing and digital design.
He carried out his Master Thesis work at Philips Research Laboratories, Aachen, Germany in 2002-2003. He is currently a member of the Bio-Inspired Intelligent Systems Research Group at Brunel University, West London, UK.
His research interests are in evolvable hardware, evolutionary computation, design of digital circuits and bioengineering applications.

Tatiana Kalganova received MSc degree from Belarusian State University of Informatics and Radioelectronics, Belarus in 1994 and PhD degree from Napier University, UK in 2000.
In August 2000 she has joined Electronic and Computer Engineering Department, Brunel University. Her research interests are evolvable hardware, ant colony algorithms, scalability in AI systems.
She was awarded a personal grant from the Education Ministry of the Republic of Belarus for distinctive achievements in the field of exact sciences in 1997, and a grant from the International Soros Science Education Program (ISSEP) for distinctive achievements in the field of exact sciences in 1996.

Cyrille Lambert received a diplôme d’éducation supérieure spécialisée in microelectronic engineering from Pierre et Marie Currie University, Paris, France in 2000.
After spending three years in the industry as a digital design engineer he joined in 2003 the computer science and engineering department at Brunel University, West London, UK. He is currently working toward the PhD. degree as a member of Bio-Inspired Intelligent Systems at Brunel University, West London, UK.
He carried out his Master Thesis work at the Swiss Centre for Electronics and Microtechnology, Inc., Neuchâtel, Switzerland in 1999-2000.